means, "whatever accesses" as the corresponding structure for the "accesses" means, and "whatever restores" as the corresponding structure for the "restores" means. The Federal Circuit has clearly and repeatedly held that functional claiming is improper; IBM's corresponding structure contentions simply ignore those holdings.

Since IBM did not take this exercise seriously, its borderline frivolous contentions should not be taken seriously. In light of the Federal Circuit's express directive "to avoid pure functional claiming," the Court should reject IBM's corresponding structure contentions on their face. *Aristocrat*, 521 F.3d at 1333. PSI's corresponding structure contentions, which are limited to actual structures disclosed in the specification, should be adopted.

**2. "means for decoding a program instruction specifying a register selected from said set of registers, said register pointing to a save area containing a saved program status word and saved register contents" (claim 19)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| Function: Decoding a program instruction specifying a register selected from the set of registers, the register pointing to a save area containing a saved program status word and saved register contents<br><br>Structure: Hardware, software, or any suitable combination of the two (see, e.g., Fig. 1; 102 5:4-11; or 7:21-28) configured to decode an instruction from a program executing in said problem state specifying a storage location containing a saved program status ward (see, e.g., Fig. 2A, 2B, 8:63-65, 9:5-15), and equivalents thereof. | Function: Decoding a program instruction specifying a register selected from said set of registers, said register pointing to a save area containing a saved program status word and saved register contents<br><br>Structure: No corresponding structure. |

IBM's proposed "corresponding structure" for this claim limitation is facially inadequate. It is a matter of black letter law that a corresponding structure must be *a structure*. *See* 35 U.S.C. § 112; *Default Proof Credit Card Sys. v. Home Depot U.S.A., Inc.*, 412 F.3d 1291, 1298 (Fed.

Cir. 2005). What IBM has proposed, however, is not a structure. *Everything* in a computer is made up of hardware or software or a combination of the two. Thus, saying that the corresponding structure is any "suitable" combination of hardware and software that performs the claimed function tells you nothing about what the corresponding structure is other than that it is part of a computer system. To put it another way, under IBM's proposal the corresponding structure is *any* part of a computer system that performs the relevant function.

As noted above, the Federal Circuit has repeatedly held that this type of claiming is not permitted, particularly in the §112 ¶ 6 context. As the Federal Circuit recently put it in *Aristocrat Technologies Australia Pty Ltd. v. Int'l Game Technology*:

> If the specification is not clear as to the structure that the patentee intends to correspond to the claimed function, then the patentee has not paid the price but is attempting to claim in functional terms unbounded by any reference to structure in the specification....Because general purpose computers can be programmed to perform very different tasks in very different ways, simply disclosing a computer as the structure designated to perform a particular function does not limit the scope of the claim to "the corresponding structure, material, or acts" that perform the function, as required by section 112 paragraph 6.

521 F.3d at 1333 (internal citations omitted).

Nor can IBM save its proposal by pointing to the parenthetical examples included in its definition. First, it is unclear what IBM thinks it is doing by pointing to examples. Again, it is black letter law that the corresponding structure for a means-plus-function limitation is ***structure*** that is disclosed ***in the specification*** and clearly linked to the function to which it corresponds. *Default Proof*, 412 F.3d at 398 ("A structure disclosed in the specification qualifies as 'corresponding' structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim."). IBM cannot simply point to things in the specification that are (in its view) ***examples*** of a broad category and then claim ***the broad category*** as the corresponding structure. Congress, in enacting § 112, limited IBM to the

structures that are ***actually disclosed*** and ***clearly linked*** in the specification. *Braun*, 124 F.3d at 1424.

Furthermore, none of the "examples" IBM points to are corresponding structures. For example, IBM asserts that "Fig. 2B and col. 9:5-15 of the '495 specification" provide corresponding structure because they "describe the algorithm for decoding this instruction." IBM Br. at 27. While an algorithm can qualify as a corresponding structure, the figure and passage IBM cites do not constitute corresponding structure because they do not show any algorithm ***capable of performing the relevant function***—namely, ***decoding*** an instruction. Again, it is black letter law that, to qualify as a corresponding structure, the identified structure must be capable of performing the function to which it is supposed to correspond. *Default Proof*, 412 F.3d at 1299 ("To meet the definiteness requirement, structure disclosed in the specification must be clearly linked to and ***capable of performing the function*** claimed by the means-plus-function limitation.").

When a processor receives an instruction, the instruction arrives as an undifferentiated string of digits. The first thing the processor does when it gets that instruction is ***decode*** it—that is, the processor figures out where and what the opcode is, and where and what the sub-fields are within the operand. '495 patent at 12:12-17; *see also* Patt Decl. ¶ 5. The process described in "Fig. 2B and col. 9:5-15" (IBM Br. at 27), however, cannot be corresponding structure for the decoding function because it is part of what the computer does with the information in the operand ***after*** the instruction has been decoded. As the patent puts it, "Figure 2B shows graphically the address arithmetic involved in ***using*** the operands." *Id.* at 9:5-6 (emphasis added). Because Figure 2B and the cited passage in column 9 deal with ***using*** the operands—not decoding the instruction to obtain the operands—they cannot be corresponding structure for the claimed function of "decoding" an instruction. *See Aristocrat*, 521 F.3d at 1334 (holding that "a

mathematical expression that describes the outcome of performing the function" is not adequate corresponding structure); *Default Proof*, 412 F.3d at 1298 (holding that corresponding structure "must include ***all*** structure that actually performs the recited function") (emphasis added). To put it another way, the step of ***decoding*** an instruction is a step that results in the processor obtaining the information shown in Figure 2A, while Figure 2B and its related passages describe what is done with some of the information in Figure 2A ***after*** it is obtained.

Nor can IBM rely on any of the other passages or figures cited in its proposed "construction." For example, IBM tries to rely on a passage in the specification stating that the CPU "has an instruction decoder for decoding instructions ... [that] may be implemented by any suitable combination of hardware and microcode in a manner well known in the art." IBM Br. at 26 (citing col. 7:21-28). This passage does not help IBM because all it says is that the instructions are decoded by an unspecified structure made up of some unspecified combination of hardware and microcode. This is like saying that the corresponding structure for the phrase "means for opening a window" is a "window opener made of any suitable combination of materials." IBM's language simply "is not clear as to ***the structure*** that the patentee intends to correspond to the claimed function." *Aristocrat*, 521 F.3d at 1333 (emphasis added). IBM's other "examples" are similarly inadequate:

> • The passage at col. 5:4-11 talks only about what the Resume Program instruction ***does*** (it "perform[s] the transition of control back from the interrupt-handling routine back to the point of interruption")—it says nothing about how to ***decode*** that instruction, nor does it describe any structure that can be used for ***decoding***. This passage is also inadequate for the additional and independent reason that nothing in the specification clearly links it to the decoding function.
>
> • The passage at col. 8:63-65 talks only about what the contents of the fields in the Resume Program instruction ***signify***: all we're told is that, with respect to Figure 2A, "a register specification (GR) 204 specifies a general register 110 (GRx) that contains the base address 206 of the save area 108." While we now know what "GR" specifies, we still have not been told anything about how to ***decode*** an instruction or what structure to use for the ***decoding***. This passage is also

> inadequate for the additional and independent reason that nothing in the specification clearly links it to the decoding function.
>
> • The box labeled "CPU 102" in Figure 1 cannot be the means for decoding because the CPU is intended to ***contain*** that means. Furthermore, "[i]n cases involving a computer-implemented invention in which the inventor has invoked means-plus-function claiming, this court has consistently required that the structure disclosed in the specification be more than simply a general purpose computer or microprocessor." *Aristocrat*, 521 F.3d at 1333.

In sum, IBM's "hardware, software, or any suitable combination of the two" contention amounts to pure functional claiming. As noted above, this type of claiming is improper:

> The point of the requirement that the patentee disclose particular structure in the specification and that the scope of the patent claims be limited to that structure and its equivalents is to avoid pure functional claiming.

*Id.*; *see also Medical Instrumentation*, 344 F.3d at 1211 ("If the specification is not clear as to the structure that the patentee intends to correspond to the claimed function, then the patentee has not paid the price but is attempting to claim in functional terms unbounded by any reference to structure in the specification."); *Biomedino, LLC v. Waters Techs. Corp.*, 490 F.3d 946, 948 (Fed.Cir.2007) ("[I]n return for generic claiming ability, the applicant must indicate in the specification what structure constitutes the means."). IBM's identification of "hardware, software, or any suitable combination of the two" as corresponding structure is an express attempt at functional claiming. And IBM's reliance on piecemeal "examples" in the specification that fail to actually disclose structure for decoding and that are not clearly linked to the decoding function is inadequate under section 112. *See Braun,* 124 F.3d at 1424 ("Structure disclosed in the specification is 'corresponding' structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim"); *Medtronic, Inc. v. Advanced Cardiovascular Sys., Inc.*, 248 F.3d 1303, 1313 (Fed.Cir. 2001) (citing *Braun* and holding that, where means-plus-function claim relating to medical device contained function of "connecting adjacent elements together," the "straight wire and hooks of Figures 7 and 8, and

the sutures" were not corresponding structure because these structures were not "clearly linked or associated with the function of connecting adjacent elements together"); *Omega Eng'g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1331-32 (Fed.Cir. 2003) (explaining that statements from experts cannot be used to "rewrite the patent's specification" to create a clear link where the language in the specification provides none). The Court should find that there is no corresponding structure for this limitation, and that Claim 19 is therefore invalid as indefinite under section 112 paragraphs 2 & 6. *See Aristocrat*, 521 F.3d 1328 (Fed. Cir. 2008) (affirming summary judgment of invalidity for indefiniteness where patent failed to disclose adequate corresponding structure).

**3. "means response to said decoding means for executing said instruction, said executing means comprising" (claim 19)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| Function: Executing the instruction<br><br>Structure: Hardware, software, or any suitable combination to execute an instruction (Fig.1, 102; Col. 5:4-11; Col. 6:66-7:7, or Col. 7:21-4), and equivalents thereto. | Function: Executing said instruction<br><br>Structure: No corresponding structure. |

IBM's proposal for this limitation suffers from the same deficiencies discussed in the previous section. Specifically, IBM's proposal does not identify *any* specific structure to serve as the "executing" means—instead, IBM attempts to claim ***all*** "hardware, software, or any suitable combination to execute an instruction" as corresponding structure. The ***point*** of section 112 ¶ 6, however, "is to avoid pure functional claiming." *Aristocrat*, 521 F.3d at 1333. Because Federal Circuit law bars IBM's "hardware, software, or any suitable combination to execute an instruction" contention, and because the four "examples" IBM points to fail to constitute corresponding structure for "executing said instruction," the Court should find that there is no corresponding structure for this claim element.

The only structure in the '495 specification linked to the "executing" function is the "execution unit" mentioned in column 7:

> CPU 102, which constitutes the primary instruction processing unit of system 100, may comprise one or more central processors (CPs) (not separately shown). As is conventional in the art, CPU 102 has an instruction decoder for decoding instructions being executed as well as ***an execution unit for executing the decoded instructions***.

'495 patent at 7:21-26 (emphasis added). Ironically, IBM does ***not*** identify the "execution unit" as corresponding structure, despite this linking language. The reason IBM does not do so is twofold. First, an execution unit is a piece of hardware, and the only execution units in PSI's machine are all made by Intel, which is fully cross-licensed to all of the asserted patents. Second, claim 19 requires that the means be "respons[ive] to said decoding means." *Id.* at 14:5. As discussed in the previous section, the '495 specification fails to disclose any adequate, clearly linked "decoding" means. But even assuming that the '495 specification did have adequate "decoding" means, **nothing in the specification indicates that the "execution unit" mentioned in column 7 is "responsive" to any "decoding means"**—the specification literally says ***nothing*** about how or in what way the "execution unit" responds to any decoding means. Thus, the language of claim 19 itself indicates that the only possible corresponding structure (*i.e.*, the "execution unit") cannot be correctly construed as corresponding structure for this means-plus-function limitation. *See Default Proof*, 412 F.3d at 1299 ("To meet the definiteness requirement, structure disclosed in the specification must be clearly linked to and capable of performing the function claimed by the means-plus-function limitation."). Even IBM recognizes this fact and does not identify the "execution unit" as corresponding structure.

The four citations IBM *does* identify, which run from the "execution unit," are all inadequate, too. "Fig. 1, 102" is simply a rectangle labeled "CPU," and "Col. 7:21-4," which is quoted above, does nothing more than identify this box as a general purpose processor. IBM's

third example, "col. 6:66-7:7," also does nothing more than describe this same rectangle as a "central processing unit." These citations are inadequate as a matter of law. *Aristocrat*, 521 F.3d at 1333 ("[i]n cases involving a computer-implemented invention in which the inventor has invoked means-plus-function claiming, this court has consistently required that the structure disclosed in the specification be more than simply a general purpose computer or microprocessor.").

The fourth and final citation IBM relies on, "Col. 5:4-11," which IBM also cited as an "example" for the "decoding" means, does not mention or even implicate "executing." Instead, that passage is a very general description of "the present invention, a new instruction, referred to herein as Resume Program (RP)." '495 patent at 5:4-11. This general description is neither linked to the executing function nor even remotely capable of communicating any structure for "executing." It, too, fails to constitute adequate corresponding structure.

The Court should find that there is no corresponding structure for this limitation, and that Claim 19 is therefore invalid as indefinite under section 112 paragraphs 2 & 6. *See Aristocrat*, 521 F.3d 1328 (Fed. Cir. 2008) (affirming summary judgment of invalidity for indefiniteness where patent failed to disclose adequate corresponding structure).

**4. "means for accessing said save area using the contents of the register specified by said program instruction" (claim 9)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| Function: Accessing the save area using the contents of the register specified by the program instruction.<br><br>Structure : Hardware, software, or any suitable combination of the two (Fig. 1; 102 ; Col. 7:21- 28; or 5:4-11) that is configured to access the save area using the contents of the register specified by the program instruction (Fig. 2A; Fig. 2B, Fig. 3A; Fig. 6, steps 601, 602, 604, 606; Col. 8:63-65; Cal. 9:5-22; Col. 10:42-45; Col. 10:46-50; Col. 10:54-59; or Col. 10:62- 66), and equivalents thereof | Function: Accessing said save area using the contents of the register specified by said program instruction.<br><br>Structure: No corresponding structure. |

IBM's proposal for this provision suffers from the same problem discussed with respect to the previous two limitations—namely, it purports to claim as the corresponding structure anything and everything in a computer system that performs the claimed function. This is improper. *Default Proof*, 412 F.3d at 1298 ("A structure disclosed in the specification qualifies as 'corresponding' structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim."); *Aristocrat*, 521 F.3d at 1333 ("The point of the requirement that the patentee disclose particular structure in the specification and that the scope of the patent claims be limited to that structure and its equivalents is to avoid pure functional claiming."). Furthermore, none of the "examples" embedded in IBM's contention constitute corresponding structure.

The first three examples IBM identifies are "Fig. 1, 102," "Col. 7:21-28," and "Col. 5:4-11." These three citations should seem familiar by now: IBM identified these same three citations as corresponding structure "examples" for the last two means-plus-function claim elements (and it cites them as examples again for the fourth means-plus-function claim element discussed in the next section). Box 102 in Figure 1 is nothing more than a rectangle labeled "CPU." The passage at col. 7:21-28 does nothing more than tell us that this box is an

"instruction processing unit" that has an "instruction decoder" and an "execution unit," both of which "may be implemented by any suitable combination of hardware and microcode in a manner well known in the art." But all computers have "instruction processing units" that contain "instruction decoders" and "execution units" which can be "implemented by any suitable combination of hardware and microcode in a manner well known in the art." In short, column 7 does nothing more than tell us that box 102 is a general purpose microprocessor. Box 102 and column 7 therefore cannot constitute adequate corresponding structure as a matter of law. *Aristocrat*, 521 F.3d at 1333 ("[i]n cases involving a computer-implemented invention in which the inventor has invoked means-plus-function claiming, this court has consistently required that the structure disclosed in the specification be more than simply a general purpose computer or microprocessor.").

The passage at 5:4-11 is inadequate because all it says is that the Resume Program ("RP") instruction can be performed at "either the microcode or the hardware level," and that it is "defined to perform the transition of control from the interrupt-handling routine back to the point of interruption." Thus nothing in this passage either identifies a specific structure or links that structure with the function of "accessing a save area."

Fig. 2A is a bit map of the fields of the RP instruction. It identifies what the names of the fields are, and where they are in the RP instruction. As the '495 specification puts it, "Fig. 2A depicts the format of a general Resume Program instruction of this invention." '495 patent at 6:42-43. Thus, nothing in this figure shows a ***structure*** capable of "accessing a save area," nor does anything in the patent link this figure to that function.

Moving on to Figure 2B, the specification tells us that "Fig. 2B shows the address arithmetic involved in executing the instruction of Fig. 2A." *Id.* at 6:45-46. In other words, figure 2B shows an algorithm for calculating the address ***at which the memory should be***

***accessed***. Figure 2B fails as corresponding structure because the memory address that should be accessed is not the same as a structure for ***actually accessing*** that address. Even IBM recognizes that calculating the address of the save area and accessing the save area are two different things. *See* IBM Br. at 30 ("Once the save area address is determined the save area can be accessed…); *see also* '520 Patent, Fig. 8 (showing that calculating the address is a different step in the process of emulation than the "perform memory access" step); *and* Patt. Decl. at ¶ 39. Thus, Figure 2B cannot be corresponding structure for this limitation.

Fig. 3A also fails to provide corresponding structure. The specification states that "Fig 3A depicts a Resume Program instruction for an s/390 embodiment described herein." '495 patent at 6:48-49. Again, however, a drawing of the fields of the Resume Program instruction does not provide a corresponding structure for ***accessing*** the memory. Instead, it is just a map of where various information is in the instruction—information that the (unspecified) structure would use when it performed the claimed access.

The next example is "Fig. 6, steps 601, 602, 604, 606." This series of steps does not tell you how to "access the save area" or provide any structure for doing so. Instead, Figure 6 simply ***assumes*** that the access happens. This is made clear in step 603, which IBM omits from its example:

> • Step 601: This step describes how to calculate the "real address" of the save area. It does not disclose any structure for "accessing the save area."
>
> • Step 602: This step describes how to calculate the "address of replacement GPR content in the save area." Again, there is no disclosure of any structure for "accessing the save area."
>
> • Step 603: This step states the following: "REPLACE CONTENT OF SPECIFIED GPR WITH THE CONTENT OF ADDRESSED SAVE AREA LOCATION." Critically, this step assumes that the save area "contents" have already been retrieved, and therefore that ***the save area has already been accessed***.

> • Step 604: This step describes how to calculate the "address of AR replacement content in the save area." Again, there is no disclosure of any structure for "accessing the save area."
>
> • Step 606: Finally, this step describes how to calculate the "address of PSW replacement content in the save area." Again, there is no disclosure of any structure for "accessing the save area."

Thus, rather than disclose an algorithm for ***accessing*** the save area, Figure 6 discloses an algorithm for calculating the location within the save area that should be accessed and which ***makes use*** of the information therein after the access has taken place.

Notably, IBM unwittingly acknowledges that Figure 6 cannot be corresponding structure for "accessing a save area" when it argues that step 603 in Figure 6 "illustrates the means for restoring." IBM Br. at 30-31. Claim 19 establishes that the means for "restoring" is ***separate and distinct from*** the means for "accessing," and that they are two *different* functions. '495 patent at 13:52-14:16. IBM is *correct* that step 603 shows how to ***restore*** part of the program context once the context information has been obtained from the memory location. But this restoration happens ***after*** the "means for accessing" has done its work.

The next example, col. 8:63-65, says only the following: "In the logical format 200 an operation code (OPCODE) 202 identifies the instruction as an RP instruction; a register specification (GR) 204 specifies a general register 110 (GRx) that contains the base address 206 of the save area 108." Like Figure 6, this passage talks only about how to calculate the area in memory that ***should be accessed***; it says nothing about how to actually perform that access and therefore under *Aristocrat* it cannot be corresponding structure. *See Aristocrat*, 521 F.3d at 1334 (holding that "language [that] simply describes the function to be performed" is insufficient to constitute corresponding structure under § 112 ¶ 6).

The next example, col. 9:5-22, discusses the fact that "Fig 2B shows graphically the address arithmetic involved in using the operands 204-212 … to generate the beginning address"

of the memory location that should be accessed. But, again, calculations that tell you ***what*** to access cannot constitute a structure for ***actually performing that access***.

The passage at 10:42-45 fails as corresponding structure for the same reason. All it says is that after the RP instruction has been decoded the CPU "obtains the real address … of the save area" by performing the disclosed calculations. But, again, figuring out ***what*** should be accessed is an entirely different thing from ***actually performing that access***.

The passage at 10:46-50 does nothing more than the passages described above. The most illuminating thing it says is that the address that has been calculated "is then used to replace the content of the general register GRx with the saved content at the addressed save area location (step 603)." But again, nothing in this passage identifies a structure (or an algorithm) for ***actually performing the access*** needed to obtain the "saved content at the addressed save area location." Instead, this passage just assumes (like Figure 6) that that access happens through some unspecified mechanism in the background. It cannot, therefore, constitute a corresponding structure.

The passages at 10:54-59 and 10:62- 66 suffer from the same problem. They describe the calculation of various addresses in memory and the use that is made of the information that is obtained from those memory locations. But they say ***nothing*** about how those accesses are achieved, and they do not identify any structure capable of performing such an access. IBM does not appear to seriously dispute this point, since later in its brief it cites these passages as evidence that "the specification also describes the steps of using the save area address combined with the offset fields to ***calculate*** the locations of the PSW, general register and access register within the save area." IBM Br. at 30 (emphasis added). Again, ***calculating*** the portion of the save area that should be accessed is a precursor to but not the same thing as ***actually accessing*** that location. This passage cannot constitute corresponding structure.

Moreover, even if the Court were to conclude that calculating the relevant address within the save area is *part* of accessing that save area, that calculation is clearly not the *entirety* of the function—just as the function of "accessing a locker at the gym" involves more than figuring out where the locker is located. *See* Patt Decl. ¶ 39. It is black letter law that a structure cannot qualify as corresponding structure unless it is capable of performing ***all*** of the functions to which it is supposed to correspond. *Default Proof*, 412 F.3d at 1298 (Fed. Cir. 2005) ("While the corresponding structure need not include all things necessary to enable the claimed invention to work, it must include all structure that actually performs the recited function."). Thus, even if the Court were to conclude that calculating the addresses within the save area is *part* of accessing the save area, the passages pointed to by IBM cannot be corresponding structure for this claim limitation.

**5. "means for restoring said program status word and said register from the saved program status word and saved register contents contained in said save area to resume execution at the instruction address contained in said saved program status word with the program context defined by said saved program status word and saved register contents" (claim 19)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| Function: Restoring the program status word and the register from the saved program status word and saved register contents contained in the save area to resume execution at the instruction address contained in the saved program status word with the program context defined by said saved program status word and saved register contents.<br><br>Structure: Hardware, software, or any suitable combination of the two (Fig. 1; 102; Col. 7:21-28; or 5:4-11) that is configured to restore the program status word and the register from the saved program status word and saved register contents contained in the same area (Fig. 8, step 810; Fig. 6, steps 603, 605, or 607; Col. 9:58-64; Col. 10:50-53; Col. 10:59-61; or Col. 10:66-11:2), and equivalents thereof. | Function: Restoring said program status word and said register from the saved program status word and saved register contents contained in said save area to resume execution at the instruction address contained in said saved program status word with the program context defined by said saved program status ward and saved register contents.<br><br>Structure : Fig. 6. |

IBM is confused about PSI's contention concerning this claim element: IBM both notes that PSI has identified Figure 6 as the corresponding structure for this claim element (IBM Br. at 30) and, at the same time, asserts on the very next page that "PSI again erroneously asserts that there is no corresponding structure for this element." *Id.* at 31. In any case, IBM's brief ***concedes*** that Figure 6 is the proper corresponding structure for this claim limitation. As IBM puts it:

> Steps 603, 605 and 607 in Figure 6…illustrate the means for restoring the PSW and the registers once their storage locations have been calculated within the save area….The '495 specification also explains these steps…in such a way that a person of ordinary skill in the art could implement the algorithm.

*Id.* Thus, the parties apparently *agree* that Figure 6 is (1) an algorithm (2) the corresponding structure for the "restoring…" function. And, indeed, a close look at Figure 6 shows that it discloses steps for calculating the location of the information needed to restore the program context (such as steps 601, 603 and 604) and other steps for ***restoring*** the program context by loading context information it into the relevant registers in the CPU after it has been obtained from the memory (such as 603 and 605).

But IBM's construction goes off the rails when it suggests that the corresponding structure is "hardware, software, ***or any suitable combination of the two*** that is configured to restore the program status word." IBM Br. at 50. Such a construction obliterates the idea of a corresponding structure by bringing within its scope ***any*** structure that performs the recited function—regardless of whether that structure is clearly linked to the function or even ***exists*** in the specification. The whole "point of the requirement that the patentee disclose particular structure in the specification…is to avoid pure functional claiming." *Aristocrat*, 521 F.3d at 1333. IBM's proposed corresponding structure must be rejected because it is not corresponding structure at all—it is pure functional claiming unbounded by anything in the '495 patent.

## B. The '789 "Time Stamp" Patent

Mainframe computers are comprised of a plurality of partitions, each of which can act independently. These partitions "typically...include or have access to timing facilities that provide date and time of day information." '789 patent at 1:28-30. This timing information can, in turn, be used for a variety of purposes, depending on the computer and the circumstances.

The '789 patent discusses one way of creating a "unique sequence value usable within a computing environment." *Id.* 2:15-16. In particular, the specification discloses a way of creating a "unique sequence value" by joining together "timing information including at least one of time-of-day information and date information; and...selected information usable in making the sequence value unique across a plurality of operating system images on one ore more processors of the computing environment." *Id.* at 2:16-23. And this is *precisely* what the initially submitted claims attempted to claim as the invention. *See* 1/23/2002 Office Action (Ex. 15) at 2. That is, IBM originally tried to patent the idea of putting together in one "sequence value" timing information and information that could be used to distinguish between simultaneous events occurring on different processors.

But the idea of creating unique sequence values by putting together timing information and a processor identifier was not a new idea. *Id.* (rejecting claims as anticipated by *Frey*). Thus, to overcome the prior art, IBM had to restrict the scope of the claims in various ways. One of the ways in which IBM restricted its claims was by making the sequence value "usable as a current time of day clock value in real-time processing." '789 patent at 15:28-29, 18:5-6. Exactly how IBM narrowed its claims by adding this restriction forms the first dispute between the parties.

The other disputes between the parties relate to the identification of corresponding structure for certain means-plus-function claim elements in claim 33.

### 1. "usable as a current time of day clock value in real-time processing" (claim 1 and 33)

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| the sequence value reflects the actual time at which the time value was requested, and is usable as a sequential timestamp where later requests always result in larger values | representing the actual time of day at which the value can be used by a program |

IBM's construction attempts to take back scope that IBM expressly abandoned during the prosecution of the '789 patent. This contravenes established claim construction principles. *See Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 535 U.S. 722, 734 (2002) ("A rejection indicates that the patent examiner does not believe the original claim could be patented. While the patentee has the right to appeal, his decision to forgo an appeal and submit an amended claim is taken as a concession that the invention as patented does not reach as far as the original claim.").

In particular, IBM is trying to convince the Court to construe the disputed phrase as meaning that the sequence value can act as a "timestamp"—*i.e.*, as a value that can be used at a later time to determine when an event happened. Indeed, IBM uses this precise word "timestamp" in its proposed definition. *See also* IBM Br. at 35 (arguing that the purpose is "to time-stamp a record with the then-current time of day").

But IBM tried this ***exact argument*** with the patent examiner, and the patent examiner emphatically rejected it. In its June 24, 2002 Response to Office Action, IBM responded to the examiner's prior art rejection by *amending* its claims to add the phrase "said sequence value representing a timestamp." *See* 6/24/2202 Response to Office Action (Ex. 16) at 21-29. As IBM described it, "timestamps are monotonically increasing in value in that two different timestamps resulting from two instructions either on the same CPU or different CPUs correctly imply the sequence of execution of the two instructions." *Id* at 15-16. In other words, as IBM used the term, a "timestamp" is something that always increases in value (*i.e.*, it is monotonic), such that

when you go back and compare two of them you can tell which one was created first. IBM then went on to argue that its invention was patentable over the prior art because the prior art did not teach the idea of using the disclosed sequence values as a monotonic "timestamp." *Id.* at 16-20.

But the examiner came back and ***rejected*** this argument, noting both that "the limitation 'timestamp'...is not disclosed and defined in the specification" and that "each of the unique values" in the cited prior art "is a time stamped value." *See* 9/26/2002 Office Action (Ex. 17) at 8-9. Indeed, the examiner noted that these prior art time stamped values were "monotonically increasing sequence values because the value is monotonically increased by the ever increasing time value." *Id.*

Faced with this rejection, IBM ***amended*** its claim to ***remove*** the claim to a "timestamp" and instead inserted a part of the currently disputed term as a limitation—namely, that the sequence value "is useable as a current time of day value." *See* 2/10/2003 Response to Office Action (Ex. 18) at 16-18. IBM argued that this *new* restriction made the claims patentably distinct as compared to the cited prior art, in which "[t]he only requirement is that the values be unique...***the values can be created at any time and then stored for later use***." *Id.* at 11-12. Thus, IBM restricted the scope of its claims by narrowing them to only cover values that could be used as the ***current*** time of day, in order to distinguish it from the prior art teaching of a value that could be produced at any time and used later.

Although the above history alone bars IBM's proposed construction, after the amendment just discussed the examiner ***again*** rejected IBM's claims based on the prior art, this time citing *Wanish*. *See* 3/19/2003 Office Action (Ex. 19) at 5. IBM responded to this second rejection by arguing that its claims were patentably distinct because "in Wanish there is no requirement that the created identifier represent the current time of day, and that the identifier be usable as a time of day value." *See* 7/10/2003 Reply to Office Action (Ex. 20) at 3. The examiner rejected this

argument, noting that the sequence value disclosed in Wanish "***indicates the current time of day value at the time of the creation***." *See* 9/29/2003 Office Action (Ex. 21) at 5.

Faced with this rejection, IBM ***further limited*** its claims by specifying that the sequence values were "usable as a current time of day clock value ***in real-time processing***." *See* 3/1/2004 Preliminary Amendment (Ex. 22) at 2-9. As IBM explained it, this amendment distinguished its claims from Wanish, in which "[t]he extracting of information [from the sequence value] takes time, and thus, cannot be used for real-time processing." *See* 1/6/2004 Response to Office Action (Ex. 23) at 11. In other words, faced with a rejection based on the fact that the sequence value in Wanish reflects the time at which the value was requested, IBM choose to ***narrow*** its claims to sequence values that can be ***used*** as the ***current*** time ***in real-time processing***—and clarified that "real-time" was synonymous with avoiding latency. *Id.*

This history is fatal to IBM's proposed construction. Under IBM's proposal, the claim limitation covers any sequence value that (1) reflects the time it was created, (2) can be used as a timestamp and (3) is monotonic. But IBM was faced with prior art that had all three qualities and, in response, narrowed its claims to sequences that can be used to show the current time in real time processing. IBM's proposed construction is therefore barred by the prosecution history. *See Festo*, 535 U.S. at 734 ("A rejection indicates that the patent examiner does not believe the original claim could be patented. While the patentee has the right to appeal, his decision to forgo an appeal and submit an amended claim is taken as a concession that the invention as patented does not reach as far as the original claim."); *Seachange Int'l, Inc. v. C-Cor Inc.*, 413 F.3d 1361, 1373 (Fed. Cir. 2005) ("Where an applicant argues that a claim possesses a feature that the prior art does not possess in order to overcome a prior art rejection, the argument may serve to narrow the scope of otherwise broad claim language.").

Moreover, because IBM is trying to gain back scope that it gave up during prosecution its proposed construction is inconsistent with the plain meaning of the disputed phrase. Specifically, IBM asks the Court to interpret "current" to mean "reflects the actual time at which the time value was requested." This is improper because it turns the word "current" into a synonym for "past."

Furthermore, IBM's own dictionary defines "real-time processing" as "the manipulation of data that are required or generated by some process ***while the process is in operation.***" *IBM Dictionary of Computing* at 559. Thus the sequence value described in the disputed phrase ("useable as a current time of day clock value in real-time processing") is one that can be used to indicate the ***current*** time ***while a process is in operation.*** This is exactly what PSI has proposed—that the value represents the "time of day at which the value can be used by a program." PSI's construction comports *exactly* with both (1) what the extrinsic evidence says the terms mean and (2) the significant amendments in the file history.

Recognizing the weakness of its proposal (particularly in light of the prosecution history), IBM makes the specious technical argument that PSI's construction is "nonsensical" because it requires a "sequence value [that] can only be used by a program at the actual time of day reflected in that value. Under PSI's construction a sequence value with a 10:02AM time value could only be used by a program at 10:02." IBM Br. at 35. That, however, is ***exactly*** what the claim language requires, and it makes perfect technical sense. In the first place, IBM's own dictionary makes clear that calculating values in real time is useful because "usually the results are used to influence the process, and perhaps related processes while it is occurring." *IBM Dictionary of Computing* at 559.

In addition, IBM ignores the issue of *precision.* If a clock counts in one minute increments, the time 10:02AM is valid for an entire minute. Lots of things can happen during

that time, and for *all* of them, 10:02 can be used to indicate the *current* time. The same thing is true in a processor. As the '789 puts it, to ensure predictable resolution a clock "should advance at least once during a time equal to, for instance, 10 average instructions." '789 at 1:60-62. Thus, the sequence value called for in the claims can be used to show the current time in real-time processing until the clock advances – which can be ten or more instructions after it is generated. Thus, there is nothing even marginally "nonsensical" about PSI's proposed construction. Indeed, in light of the prosecution history and the express claim terms at issue, PSI's construction is the only one that makes sense.

**2. § 112(6) Limitations**

Claim 33 of the '789 patent contains a number of means plus function limitations that are related to one another. Because the issues are essentially the same for each disputed limitation, they are treated together below, rather than on a limitation by limitation basis.

<table>
<tr><th>Claim Limitation</th><th>IBM's Proposed Construction</th><th>PSI's Proposed Construction</th></tr>
<tr><td rowspan="2">"at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable within a computing environment, the computer readable program code means in said article of manufacture comprising" (claim 33)</td><td colspan="2">Agreed Function: Causing the generating of unique sequence values usable within a computing environment.</td></tr>
<tr><td>Structure: An <u>article of manufacture</u> (see, e.g., Col. 14:61-15:4) having computer usable media for causing the generating of unique sequence values usable within a computing environment (see, e. g., Fig. 9; Col. 8:62-67; Fig. 10, step 1010, 1002 ; Col. 9:1-5; Col. 9:13-15; Col. 11:23-35; <u>or</u> Col. 11:61-12:6), and equivalents thereof.</td><td>Structure:<br><br>No corresponding structure.</td></tr>
</table>

<table>
<tr><td rowspan="3">(i) "computer readable program means for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information" (claim 33)</td><td colspan="2">Agreed Function: Causing a computer to provide as one part of a sequence value timing information comprising at least one time-of-day information and date information.</td></tr>
<tr><td>Structure: An <u>article of manufacture</u> (Col. 14:61- 15:4) having computer usable media for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information (Fig. 9; Col. 8:62-67; Fig. 10, step 1002; <u>or</u> Col. 9:1-5), and equivalents thereof.</td><td>Structure:<br><br>No corresponding structure.</td></tr>
<tr><td></td><td></td></tr>
<tr><td rowspan="2">(ii) "computer readable program means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment" (claim 33)</td><td>Function: Causing a computer to include as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment.</td><td>Function: Causing a computer to include as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment.</td></tr>
<tr><td>Structure: An <u>article of manufacture</u> (Col. 14:61-15: 4) having computer usable media for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operation system images on one or more processors of said computing environment (Fig. 9; Col. 8:62-67; Fig. 10, step 1010; Col. 9:13-15; Col. 11:23-35; <u>or</u> Col. 11:61-12:6), and equivalents thereof.</td><td>Structure:<br><br>No corresponding structure.</td></tr>
</table>

**a. Function**

As an initial matter, the parties disagree regarding the function to be performed by limitation (ii), above. The language of the claim limitation recites:

> . . . causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment, **wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment**.

'789 patent at 17:48-18:7 (emphasis added). The bolded language, verbatim, is PSI's proposed function for this claim limitation. The function IBM wants is the following:

> Causing a computer to include as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment.

IBM Br. at 39. IBM's proposal is improper because it omits the claim's express requirement that "said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment." This "wherein" clause that IBM attempts to omit provides an important limitation regarding the "sequence value" referred to in the remainder of the claimed function. Without it, and under IBM's proposal, every "computer readable program means for causing" the inclusion of the "selected information" would fall within the claims, whether or not the resulting sequence value was "usable as a current time of day clock value in real-time processing by one or more processors of the computing environment." But there is no basis for ignoring this express limitation of the claim language.

In support of its desire to ignore the "wherein" clause, IBM cites *Lockheed Martin Corp. v. Space Systems/Loral, Inc.*, 324 F.3d 1308, 1319 (Fed. Cir. 2003). But the *Lockheed Martin* case deals with a **"whereby"** clause, not a "wherein" clause. As the Federal Circuit noted in *Griffin v. Bertina*, 285 F.3d 1029 (Fed. Cir. 2002), the two words are different parts of speech:

"'wherein' is an adverb and 'whereby' is a conjunction." *Id.* at 1034. The Federal Circuit held in *Griffin* that the "wherein" clauses at issue "relate back to and clarify what is required by the count," "express[ing] the inventive discovery" of the patent. *Id.* at 1033-34. Similarly, in *Intergraph Hardware Technologies Co. v. Toshiba Corp.*, 508 F. Supp. 2d 752 (N.D. Cal. 2007), the court, faced with an identical argument about a "wherein" clause, looked to "whether the 'wherein' clause...expresses an inventive component or merely the result of the delineated limitations." *Id.* at 769. Here, as discussed above, the "usable as a current time of day clock value in real-time processing" limitation is not the necessary ***result*** of the remaining limitations. Instead, it recites an ***additional*** capability of the claimed sequence value. That this limitation constitutes "an inventive component" is shown by the prosecution history discussed above, in which the "usable as a current time of day clock value in real-time processing" limitation was a hotly contended issue. IBM cannot now simply omit this unwanted limitation. PSI's proposed "function" should be adopted.

**b. Structure**

As occurred with the '495 patent, IBM's corresponding structure contentions are seriously flawed: they literally include every structure known to man that performs the stated functions. Specifically, IBM attempts to claim ***all*** "articles of manufacture" containing any software that causes a general purpose computer to perform the function recited in the limitation. But, under *Aristocrat*, a means plus function limitation cannot be construed in this manner:

> Because general purpose computers can be programmed to perform very different tasks in very different ways, simply disclosing a computer as the structure designated to perform a particular function does not limit the scope of the claim ... as required by section 112 paragraph 6.

*Aristocrat*, 521 F.3d at 1330.

Nor can IBM point to its parenthetical "examples" as a viable alternative to its unbounded proposal. As in several other of IBM's asserted patents, the supposed "examples" of

corresponding structure that IBM identifies are actually all ***the same single generic boilerplate description*** of computer software:

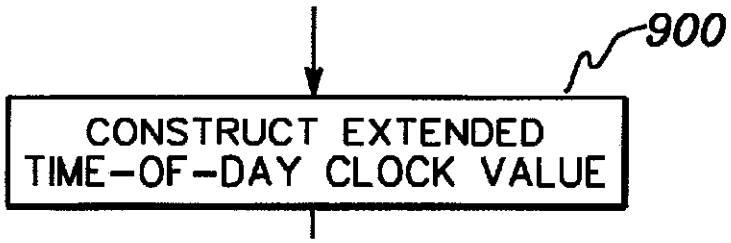
> The present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The article of manufacture can be included as a part of a computer system or sold separately.
>
> Additionally, at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

'789 patent at 14:61-15:4. Notably, this text is so generic that it is ***identical*** to the text in the '812 patent that IBM identifies as the "structure" for the means-plus-function limitations in claim 11 of that patent. IBM Br. at 71-76 (citing the identical passage at '812 patent at 10:46-57 as corresponding structure for 4 means-plus-function limitations in the '812 patent). **Indeed, IBM has used this *identical* generic language in *over 90* U.S. patents and patent applications**.[20] As in the '812 patent, this generic description of software on tangible media does not constitute corresponding structure because it does not disclose ***what*** is recorded on that media that actually is capable of performing the function—all that is mentioned is the generic "computer readable program code means for providing and facilitating the capabilities of the present invention." Code "for providing and facilitating the capabilities of the present invention" is, however, no more of a structure than "appropriate programming," the purported structure expressly rejected by the Federal Circuit in *Aristocrat*. *See* 521 F.3d 1328 (Fed. Cir. 2008).

The other passages cited by IBM are similarly inadequate. First, by their express terms, IBM's "examples" attempt to identify no fewer than four ***alternatives*** for corresponding structure: "e.g., Fig. 9; Col. 8:62-67; Fig. 10, step 1002; or Col. 9:1-5." Each of these citations is

[20] *See* http://www.google.com/patents?q=%22The+present+invention+can+be+included,+for+example,+in+an+article+of+manufacture%22&lr=&start=0&scoring=2 (providing links to more than 90 other IBM patents and pending patent applications containing identical language).

insufficient to provide an algorithm for the stated function. For example, one step of the process illustrated in Figure 9 is:




This step tells the reader nothing about how to construct such a value—and certainly does not provide any algorithm for doing so.

The other three "examples" are similarly inadequate:

> • Col. 8:62-67 does nothing more than ***describe*** part of the function: it says "an extended time of day clock value is constructed by filling in representation 400, STEP 900. In one embodiment, the value is constructed, as described below with reference to Fig. 10." All we are told is that the "time of day clock value" ***is*** constructed—we're told nothing about ***how*** or ***with what*** it is constructed. *Aristocrat*, 521 F.3d at 1333 (holding that "language [that] simply describes the function to be performed" is insufficient to constitute corresponding structure under § 112 ¶ 6).
>
> • Figure 10 step 1002 says only to "set bits 8 to S+8 to value of running clock." But this, too, is inadequate structure "for causing a computer to provide as one part of a sequence value timing information comprising at least one time-of-day information and date information." How does one retrieve the value of the running clock? Where is the value of the running clock located? *See Default Proof*, 412 F.3d at 1298 ("While the corresponding structure need not include all things necessary to enable the claimed invention to work, it must include all structure that actually performs the recited function".).
>
> • Finally, col. 9:1-5 does nothing more than mention step 1002 and state that the running clock could be located either in the CPU or outside of the CPU. Again, we are left with insufficient structure to actually perform the recited function.

The best argument in defense of the claims is that the above passages, *taken together* (and not as mere examples or disjoint options, as IBM uses them), might provide a structure corresponding to some of the stated "causing" functions. But, even if IBM had made that argument (it didn't—instead it opted for trying to claim "any article of manufacture…"), the cited passages cannot form an algorithm ***capable*** of performing the claimed function, because at most they provide

information only as to ***some*** of the tasks required for the function (*e.g.*, setting bits to the value of the running clock) but not others (*e.g.*, retrieving the value). *Default Proof*, 412 F.3d at 1298 ("While the corresponding structure need not include all things necessary to enable the claimed invention to work, it must include all structure that actually performs the recited function".).

## V. THE EMULATION PATENTS

### A. The '520 "Address Translation" Patent

Computers consist of both hardware and software. In order to function properly, hardware and software must be compatible. Compatibility is achieved by means of a specification that defines a set of conventions that both the hardware and the software observe. As noted above, this specification is known as an Instruction Set Architecture or "ISA." Patt Decl. ¶ 12. Among other things, an ISA contains a list of the operations that the hardware can perform, the codes (called "op-codes") that will trigger the hardware to perform those operations, and the format in which instructions must be put together. *Id.*

Emulation is the practice of using a computer with one ISA (*e.g.*, an Intel-based system) to imitate a computer that has a different ISA (*e.g.*, an IBM-based system) "so that the imitating system accepts the same data, executes the same programs, and achieves the same results as the imitated system." Joint Statement at Ex. A p. 3. Emulation is not new: computer scientists have been emulating computers with different architectures since the 1960s. Furthermore, while individual techniques vary,[21] one thing is universally true in computer emulation: all emulation systems must have a mechanism for ***translating*** the incompatible programs into instructions that the imitating (or "host") computer can understand.[22] This translating mechanism is necessary because the host computer is ***incapable*** of executing instructions that are not written in its ISA.

---

[21] PSI, for example, has a patent on certain aspects of its unique emulation system. *See* U.S. Pat. No. 7,092,869 (Ex. 26).

[22] *See, e.g.*, '520 at 262-64 (it is an "object of the preset invention to provide a method and system for address ***translation*** during emulation…"); and '261 at 3:62-65 ("The subject invention provides a program ***translation*** and

The '520 patent describes a system and method for performing a specific type of translation called *address translation*. All computer systems store information in memory and designate different locations in that memory by using "memory addresses." Patt Decl. ¶ 39. Furthermore, as one might expect, computers with different ISAs handle memory addressing in different ways. Thus, for an emulation system to work properly, it must be able to *translate* the memory addresses used by software written for one ISA into the memory addresses used by hardware designed to implement a different ISA.

The '520 patent discloses one way of performing that translation. Specifically, the '520 claims a system and method for performing this translation in two steps, first from a "guest logical address into a guest real address," and then "thereafter...into a native physical address." '520 at 16:23-27 (claim 1). All this means is that the translation process starts with the address format used by *computer programs* in the emulated ISA (the "guest *logical* address"), translates that into the address format used by *hardware* in the emulated ISA (the "guest *real* address"), and then translates the result into the format used by the *hardware* carrying out the emulation (the "native physical address").

The parties dispute the construction of "processor," "semantic routine," "instruction set," and the corresponding structure for the means-plus-function limitations in claim 9. The construction of "processor" is set forth above in Section III.

**1. "semantic routine" (claims 1, 4, 9, 12)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| A defined set or sequence of native instructions that, when executed emulates an associated guest instruction. | A defined sequence of native instructions that, when executed, emulates an associated guest instruction. |

execution system which allows the processor to execute an incompatible program coded to a different computer architecture.").

The single issue that the Court must decide with respect to this claim term is whether the native instructions that comprise a "semantic routine" can be an unordered set or whether this routine is understood to be a sequence. The intrinsic and extrinsic evidence, as well as elementary principles of computing and logic, demonstrate that a "semantic routine" is necessarily a ***sequence*** and not just a set of instructions.

It is an elementary principle of computing as well as logic that operations performed in a different order can (and typically do) yield different results. Patt Decl. ¶¶ 34-38. This principle stems from the fact that there are ***dependencies*** between the operations that a computer performs. *Id.* at ¶ 36. For example, a sequence of instructions in a simple program might tell the processor to (1) subtract the number stored in register 1 from the number stored in register 2, (2) store the result in register 3, and (3) close a circuit that will cause a beeping sound to play if the number in register 3 is negative. The user of this program obviously could get very different results if those three steps were taken out of order.

IBM tries to obscure this elementary principle with the following argument:

> For example, if one's morning routine is to do "X, Y, and Z" before heading to work, it is no less of a routine if one does "Y, X, and Z" one day and "Z, X, and Y" the next.

IBM Br. at 43. But even this simplistic argument fails, since one's morning routine has obvious dependencies—just take the case where "X=wake up," "Y=shower," and "Z=get dressed." Indeed, this simple example accurately illustrates the problem with IBM's construction: everyone in the art knows that computer instructions must be performed as a sequence (*i.e.*, in an order) and not as a set (*i.e.*, in any random order), otherwise you get unintended or nonsensical results (like showering with your clothes on). For that reason, a "routine" in the computer

context is commonly understood to be "a ***sequence*** of computer instructions for performing a particular task."[23] *See* Patt Decl. ¶¶ 34-38.

This is precisely how the term "semantic routine" is used and understood in the '520 patent. The '520 claims, for example, call for "storing in memory ***a semantic routine*** of native instructions ***from said native instruction set***..." '520 patent at 16:18-20 (emphasis added). Thus, the claims clearly differentiate the idea of a semantic routine from the broader concept of an "instruction set." Under IBM's proposed construction—where a "semantic routine" can be ***any*** defined set of instructions—this distinction disappears.

Similarly, in column 4, the specification gives an example of a "semantic routine" that emulates the incompatible instruction "ADD MEM1, MEM2, MEM3." In the emulated or "guest" system, that instruction causes the processor to add the contents of memory location 1 (MEM1) to the contents of memory location 2 (MEM2) and then store the result in memory location 3 (MEM3). The emulating or "host" system does not have a single instruction that it can execute to perform that task, so it must instead emulate the guest system by executing the following semantic routine of native instructions:

LOAD REG1, MEM1

LOAD REG2, MEM2

ADD, REG3, REG2, REG1

STORE REG3, MEM3

'520 patent at 4:55-63. This routine produces the same result as the emulated instruction but in a different way. Specifically, the host system loads the contents of memory locations 1 and 2 into registers 1 and 2, adds the contents of those registers into register 3, and then writes the contents of register 3 into memory location 3. *Id.* at 4:63-67.

[23] http://www.merriam-webster.com/cgi-bin/dictionary?book=Dictionary&va=routine (Ex. 24).

It should be obvious, however, that the steps in the semantic routine have dependencies and ***cannot be taken in any order***. Until the values are loaded into registers 1 and 2, for example, they ***cannot*** be added together into register 3. And, until they are added together into register 3, the result of the addition ***cannot*** be written into memory location 3. This semantic routine, like every other semantic routine in the intrinsic evidence, is a ***sequence*** and not a set.

To those of ordinary skill in the art, it is obvious that PSI's construction is correct and that IBM's proposal makes no sense. **The inventor of the '520 patent**, for example, testified as follows:

[redacted][24]

In other words, when asked about it in deposition, the '520 inventor explicitly rejected IBM's construction and agreed with PSI's position. Furthermore, PSI's expert Professor Patt had this to say about the parties' respective constructions:

> Dr. Smotherman [IBM's expert] is correct, of course, that adding 1 to 2, then adding 3 gives the same result as adding 3 to 2, then adding 1. This is because, as a middle school math teacher would say, addition displays the associative property. But, as the example given in column 4 of the '520 patent makes clear, even the process of adding numbers requires a *sequence* of instructions. In particular, the processor *must* load the data representing the numbers *before* it adds those numbers together, and it must add the numbers together *before* it stores the result. Thus, even in Dr. Smotherman's simple example, a semantic routine to emulate an instruction to add 1+2+3 would have to have an internal order and could not fairly be characterized as simply being a "set" of instructions. In short, it is universally understood in the art that computer instructions have *dependencies*, and that a routine must have an internal order. This concept is so

[24] [redacted]

> elementary that it would strike one of ordinary skill in the art as ridiculous to suggest that the instructions in a routine should not be understood to have an internal order.

Patt Decl. ¶ 36 (emphasis in original).

IBM's brief tries to take advantage of the fact that a "sequence" is a subset of the term "set." In other words, all sequences are sets (with special conditions), in the same way that all squares are rectangles (with special conditions). As a result, IBM has been able to find a couple of instances in which people have referred to a routine as a "section of code" or as "set of host instructions." IBM Br. at 43. But the fact that people sometimes fail to mention the internal relationships between the elements of a routine does not mean that those relationships are simply ***optional***. Thus, for example, the reason the Microsoft Dictionary defines a "routine" as a "section of code" is because people of ordinary skill in the art ***don't need to be told*** that the code in the routine is intended to be performed in a sequence. (Ex. 30 at 456). *See also* Patt Decl. ¶ 36. Indeed, if the section of code were performed in an arbitrary order—without regard to any internal *sequence*—the code would fail to accomplish any useful function. *Id.*

IBM will likely argue in reply that PSI's construction is too narrow because the elements of a routine do not ***always*** have to be performed in the same fixed order. This argument has no more merit than IBM's morning-routine argument. For example, it is true that in the semantic routine shown in column 4 and discussed above that either of the two load instructions could be performed first without changing the result (*i.e.*, it doesn't matter if you load register 1 then register 2 or vice versa). '520 patent at 4:55-63. But that fact does not detract from the argument in favor of PSI's construction because there is nothing in the concept of a ***sequence*** that ***requires*** one and only one fixed internal order. For example, "(A or B) then C then D" is still a ***sequence*** even though, like the example from column 4, it does not have a single fixed order.

In contrast, abstracting out the requirement for ***any*** internal order, as IBM proposes, destroys the fundamental concept of a routine and makes it a mere synonym for the words "group" and "set." Such a result might help IBM's infringement case, but it conflicts with the meaning of the term as used in the '520 patent and as understood by those of ordinary skill in the art.

**2. "instruction set" (claims 1, 9)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| The complete set of operations of the instructions of ***a computer architecture*** together with the types of meanings that can be attributed to their operands. | The complete set of the operations of the instructions of ***a computer*** together with a description of the types of meanings that can be attributed to their operands. |

The difference between the parties' constructions is subtle: IBM would like "instruction set" to refer to the complete set of operations of a ***computer architecture***, while PSI contends that "instruction set" should be construed to refer to the complete set of operations of a ***computer***. The question that the Court actually has to answer as a result of this difference is the following: can one computer have two different instruction sets? The intrinsic and extrinsic evidence overwhelmingly establish that computers have only one "instruction set" and that PSI's construction, which is how the term is used in the claims, should be adopted.

PSI does not dispute that "instruction set" can refer to the complete set of operations of the instructions of ***a computer architecture***. For example, the specification mentions the PowerPC and the Intel x86 computer architectures, and those two computer architectures each consist of a particular set of conventions, including but not limited to a particular instruction set. However, **it is just as accurate** to say that a particular ***computer*** or processor has a particular or native "instruction set." Indeed, that is precisely how the claims at issue use the term "instruction set":

| Claim 1 | Claim 9 |
|---|---|
| 1. A method of operating ***a processor which has a native instruction set*** and emulates instructions in a guest instruction set, said method comprising:<br>storing, in memory, a series of guest instructions from said guest instruction set, said series including a guest memory access instruction that indicates a guest logical address in guest address space;<br>for each guest instruction in said series, storing in memory a semantic routine of native instructions from said native instruction set to emulate each guest instruction, said native instructions utilizing native addresses in native address space;<br>in response to receipt of said guest memory access instruction for emulation, translating said guest logical address into a guest real address and thereafter translating said guest real address into a native physical address; and<br>executing a semantic routine that emulates said guest memory access instruction utilizing said native physical address. | 9. A ***processor which has a native instruction set*** and emulates instructions in a guest instruction set, said processor comprising:<br>guest instruction storage that stores guest instruction from a guest instruction set, wherein said series includes a guest access instruction that indicates a guest logical address in guest address space;<br>semantic routine storage that stores a plurality of semantic routines of native instructions for emulating said series of guest instructions;<br>means, responsive to receipt of said guest memory access instruction for emulation, for translating said guest logical address into a guest real address and for thereafter translating said guest real address into a native physical address; and<br>means for executing a semantic routine that emulates said guest memory access instruction utilizing said native physical address. |

The specification also clearly uses "instruction set" to refer to the complete set of operations of a ***computer***. For example, the specification refers to "native" instructions as "instructions that are within the standard instruction set **of CPU 4**," and, with respect to this same exemplary computer, describes "guest" instructions as "CISC instructions or some other instruction set that is not native **to CPU 4**." '520 patent at 4:41-42, 11:4-6 (emphasis added); see also *id.* at 1:48-49 (noting that the PowerPC architecture "provides a definition of the instruction set…**for a family of computer systems**") (emphasis added). Thus, the intrinsic evidence expressly rejects IBM's exclusion of PSI's "of a computer" construction.

Because, as noted above, PSI does not disagree that the term "instruction set" can be used to refer to the complete set of operations of a computer architecture, PSI is willing to accept the following combination of the parties' constructions:

> "instruction set" (claims 1, 9): The complete set of operations of the instructions of a computer or a computer architecture together with the types of meanings that can be attributed to their operands.

Although the claims and the specification plainly use the term "instruction set" to refer to the complete set of operations of particular ***computers***, PSI does not dispute IBM's contention that, in column 1, the specification also uses "instruction set" with reference to a computer architecture. *See* IBM Br. at 41 (citing 1:45-64). Thus, PSI will accept the above combined construction.

It would be error, however, to adopt IBM's construction and exclude the use of "instruction set" as referring to the complete set of operations ***of a computer***, since that is how the term is used in both the claims and in the portions of the specification cited above. The extrinsic evidence also supports this commonly accepted meaning of the term. *See IBM Dictionary of Computing* (1994) at 346 (defining "instruction set" as "the set of instructions **of a computer**, of a programming language, or of the programming language in a programming system"); *The IEEE Standard Dictionary of Electrical and Electronic Terms* (6th ed. 1996) (Ex. 25 at 563) (defining "instruction set" as "[t]he complete set of instructions ***recognized by a given computer***...this term is considered to be synonymous with a computer's architecture.").

Notably, the only reason this dispute is being put to the Court is because of IBM's preoccupation with establishing literal infringement of ***other patents*** in this case. None of the parties' claims or defenses concerning the '520 patent turn on how the Court construes "instruction set." Instead, IBM refuses to accept PSI's construction (and the combined construction above) because IBM wants to be able to say that "the computer discussed in the

'520 patent supports *two distinct* instruction sets." IBM Br. at 42 (emphasis in original). The reason IBM wants to be able to make this statement (and seeks support for it through its construction) is that several of the other patents-in-suit claim instructions that are IBM-architecture instructions and that are therefore not within the instruction set of PSI's computers. For example, PSI's Intel-based computers cannot execute the Test FP Data Class instruction claimed in the '678 patent.

IBM acknowledges that Intel-based computers cannot execute IBM instructions. IBM Br. at 6 ("IBM instructions are not compatible with the non-IBM architecture of the Intel-based computers"); *see also id.* at 7 ("Intel-based computers...are incapable of understanding and executing IBM instructions"). However, because that fact is fatal to its literal infringement case, IBM needs to argue that a PSI computer "supports *two distinct* instruction sets" when it engages in emulation. IBM Br. at 42 (emphasis in original). But that argument has to be wrong: an Intel-architecture processor does not become an IBM-architecture processor simply because it runs an emulation software program. Instead, the emulating machine still has the same, single Intel architecture—it simply executes sequences of ***Intel*** instructions to ***emulate*** the IBM instruction set.

Thus, it is IBM's "two instruction set" argument that forces IBM to take the bizarre position that "instruction set" cannot refer to the complete set of operations of a ***computer***, despite the fact that the '520 claims and specification use the term in that way and despite the fact that the extrinsic evidence also uses the term in that way. Furthermore, the only two arguments that IBM makes to support its bizarre position both lack merit. IBM's first argument is, on its face, a non sequitur:

> **[1]** The '520 patent relates generally to a method and system for operating a processor that has a "native" instruction set and emulates instructions in a "guest" instruction set. **[2]** The parties have agreed that "native" means "pertaining to the *architecture* on which the emulator runs" and "guest" means "pertaining to the

> *architecture* that is being emulated. **[3]** Thus, at the outset, PSI's position contradicts the parties' agree construction for the terms native and guest.

IBM Br. at 41 (emphasis in original). PSI's construction simply says that "instruction set" refers to the complete set of operations of a ***computer***. Nothing about that statement is inconsistent with the facts that (1) the '520 patent uses the phrases "native instruction set" and "guest instruction set" and (2) the parties' agreed constructions for "native" and "guest" both begin with "pertaining to the architecture…". "Architecture," like "instruction set," can and frequently is used to refer to the "instruction set" or the "instruction set architecture" of a ***computer***. Indeed, IBM uses "architecture" that way in its brief: "IBM instructions are not compatible with the non-IBM ***architecture of*** the Intel-based ***computers***." IBM Br. at 6 (emphasis added). In short, the conclusion IBM tries to draw from its two premises simply does not follow.

IBM's second argument is incomprehensible:

> Under PSI's construction, the instruction set "of a ['520] computer" would have to include a combination of both instruction sets, which would render the emulator unnecessary and would therefore make no sense.

IBM Br. at 42. IBM does not explain what it means by "include a combination of both instructions sets" or why PSI's construction requires that result (whatever it means). Instead, after making the single incomprehensible assertion quoted above, IBM summarily asserts that "[a]s a result, IBM's construction…is the proper construction." *Id.* But IBM's argument makes no sense: under PSI's construction, computers embodied by the '520 patent have a ***single*** instruction set (*e.g.*, they are ***RISC*** computers emulating CISC instructions). It is *IBM* who makes the specious argument that "the computer discussed in the '520 patent supports *two distinct* instruction sets." *Id.* Ironically, PSI ***agrees*** with IBM's apparent recognition that this dual instruction-set argument "would render the emulator unnecessary and would therefore make no sense." *Id.*

In sum, the Court should adopt PSI's construction or the combined construction set forth above. IBM's construction is contrary to the intrinsic and extrinsic evidence and should be rejected.

**3. "means, responsive to receipt of said guest memory access instruction for emulation, for translating said guest logical address into a guest real address and for thereafter translating said guest real address into a native physical address" (claim 9)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| Function: Translating said guest logical address into a guest real address and far thereafter translating said guest real address into a native physical address.<br><br>Structure: A ***data processing system*** (Figs. 1, 2, **or** 3) configured to translate the guest logical address into a guest real address and thereafter to translate the guest real address into a native physical address (Figs. 7 **or** 8; column and lines: 13:52-14:45; or 14:46-15:7; **or** 15:56- 65), and equivalents thereof. | Function: Translating said guest logical address into a guest real address and for thereafter translating said guest real address into a native physical address.<br><br>Structure: Figs . 7, 8, 9. |

The parties agree that the function of this claim limitation is to perform the particular type of address translation that allegedly distinguishes the patent from the prior art.[25] The parties disagree about what corresponding structure is disclosed by the specification.

IBM's proposal, like all of its means-plus-function proposals, seeks to encompass ***any and all*** possible structures that can perform the claimed function. Thus, IBM contends that the corresponding structure should be construed as every "data processing system" that is "configured" to perform the claimed function. This kind of construction is improper. *See Aristocrat*, 521 F.3d at 1334 ("The term 'appropriate programming' simply references a computer that is programmed so that it performs the function in question, which is to say that the

[25] January 14, 1990 Response to Office Action (Ex. 27) at 2 (arguing that the claims of the '520 were patentable over the prior art because nothing therein "teaches or suggests the particular address translation recited in Claim 1.").

function is performed by a computer that is capable of performing the function."). By statute, a patentee is not entitled to ***all*** possible means of performing the function, but instead is limited to "the corresponding structure, material, or acts described in the specification and equivalents." 35 U.S.C. § 112 ¶ 6; *Atmel Corp. v. Info. Storage Devices*, 198 F.3d 1374, 1381 (Fed. Cir. 1999) ("structure supporting a means-plus-function claim under § 112, ¶ 6 must appear in the specification."). IBM's proposal should be rejected for this reason alone.

Putting aside IBM's awkward attempt to claim an abstract "data processing system" as corresponding structure, the "examples" that IBM cites from the specification to support its abstract "data processing system" are also improper, at least in part. Specifically, the general purpose processing systems in Figures 1, 2 and 3 are improper under *Aristocrat*. *Aristocrat*, 521 F.3d at 1333 (Fed. Cir. 2008) ("This court has consistently required that the structure disclosed in the specification be more than simply a general purpose computer or microprocessor....[A] computer-implemented means-plus-function term is limited to the corresponding structure disclosed in the specification and equivalents thereof, ***and the corresponding structure is the algorithm***.") (emphasis added, internal citations omitted). Instead, the only proper corresponding structure in the specification are the algorithms disclosed in Figures 7, 8 and 9. The parties ***agree*** that those algorithms actually perform the address translation in the claimed function. Thus, Figures 7, 8 and 9 should be construed as corresponding structure.

The only remaining question is whether the passages in columns 13, 14 and 15 that IBM cites as examples ("column and lines: 13:52-14:45; or 14:46-15:7; or 15:56- 65") should also be construed as corresponding structure. These three passages discuss Figures 7, 8 and 9. However, because these passages only ***describe*** the function rather than actually demonstrate how to carry it out, the corresponding structure should be limited Figures 7, 8 and 9.

**4. "means for executing a semantic routine that emulates said guest memory access instruction utilizing said native physical address"(claim 9)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| Function: Executing a semantic routine that emulates said guest memory access instruction utilizing said native physical address.<br><br>Structure: A ***data processing system*** (Figs. 1, 2, or 3) configured to execute a semantic routine that emulates the guest memory access instruction utilizing the native physical address (Fig. 8; column and lines: 11:4-15; 11:26-31; 13:60-14:25; 15:56-67), and equivalents thereof. | Function: Executing a semantic routine that emulates said guest memory access instruction utilizing said native physical address.<br><br>Structure: Figs. 7, 8, 9. |

As with its other proposed corresponding structures, IBM's construction makes the fundamental mistake of attempting to claim all possible structures that perform the claimed function. Specifically, IBM contends that the corresponding structure is any "data processing system" that is "configured" to perform the claimed function. But this kind of construction is improper. *See Aristocrat*, 521 F.3d at 1334 ("The term 'appropriate programming' simply references a computer that is programmed so that it performs the function in question, which is to say that the function is performed by a computer that is capable of performing the function."). By statute, a patentee is not entitled to ***all*** possible means of performing the function, but instead is limited to "the corresponding structure, material, or acts described in the specification and equivalents." 35 U.S.C. § 112 ¶ 6; *Atmel*, 198 F.3d at 1381 ("structure supporting a means-plus-function claim under § 112, ¶ 6 must appear in the specification."). Thus, IBM's definition is improper.

In its brief, IBM does not defend its abstract "data processing system" construction but instead points to each of the parenthetical references in its proposal as providing corresponding structure. With respect to those parenthetical references, the parties' dispute can be summarized as follows: PSI (i) objects to Figures 1, 2 and 3; (ii) agrees with IBM's remaining citations (*i.e.*,

Figure 8 and 11:4-15, 11:26-31, 13:60-14:25, 15:56-67); and (iii) contends that Figures 7 and 9 should also be included as corresponding structure. IBM's brief states that it agrees that Figure 7 should be added as corresponding structure but it continues to object to Figure 9. IBM Br. at 46. Thus, the only issue the Court needs to decide is whether Figures 1, 2, 3, and 9 should be included as corresponding structure.

IBM claims that Figures 1, 2 and 3 constitute corresponding structure because they show different views of a general purpose microprocessor, and because the specification says that semantic routines of native instructions "that are within the standard instruction set of CPU 4 are processed by CPU 4 as described above with reference to Fig 2." IBM Br. at 45 (citing 11:4-6). But, under *Aristocrat*, general purpose microprocessors cannot be corresponding structure. Instead, when an algorithm is disclosed to make a general purpose computer processor act in a specific way, the corresponding structure is the algorithm, not the processor:

> This court has consistently required that the structure disclosed in the specification be more than simply a general purpose computer or microprocessor....[A] computer-implemented means-plus-function term is limited to the corresponding structure disclosed in the specification and equivalents thereof, ***and the corresponding structure is the algorithm***.

*Aristocrat*, 521 F.3d at 1333 (Fed. Cir. 2008) (emphasis added). Thus, Figures 1, 2 and 3, which depict only general purpose processors and not any algorithms capable of performing the claimed function, are not properly part of the corresponding structure for this claim limitation.

With respect to Figure 9, IBM contends that this figure should not be construed as corresponding structure because it is corresponding structure for the "means for translating function." That argument lacks merit. First, nothing prevents a particular figure from constituting corresponding structure for more than one function. Indeed, IBM itself claims that Figure 7, Figure 8, and the same certain passages in columns 13 and 14 all constitute structure for both the "means for executing a semantic routine" function (at issue here) and the "means for

translating" function. Thus, the fact that Figure 9 is corresponding structure for the "means for translating function" does not demonstrate that it cannot also be corresponding structure for this claim limitation.

Furthermore, the specification states that Figure 9 is "a diagram of certain registers ***used in the address translation scheme illustrated in Figs 7 and 8***." '520 patent at 3:56-57. Thus, it is inconsistent for IBM to argue that Figures 7 and 8 are corresponding structure here but that Figure 9 is not—the specification demonstrates that you cannot have it both ways. Specifically, Figure 8 shows the native instructions that must be executed in order to emulate a guest memory access instruction. The steps in that routine ***include*** the translation of the memory address used by the guest instruction. *Id.* at Fig. 8. Thus, it is not possible to separate the address translation process from the emulation process—one is a subset of the other. Because the patent expressly states that Figure 9 is a diagram of the structure that performs this address translation (*id.* at 3:56-57), the Court should construe Figure 9 as part of the corresponding structure for this limitation.

**B. The '261 "Patching" Patent**

The '261 patent also deals with emulation, but focuses on a different part of the process. In particular, the '261 focuses on a way to speed up emulation by storing a set of "target routines" in memory. These target routines are incomplete semantic routines that can be completed through a process the '261 calls "patching." '261 at 8:50-57. Once they are patched, these target routines can then be executed in order to emulate a guest instruction.

By storing these incomplete "target routines" in memory the '261 design "allows an emulation that avoids the large target storage overhead" of previous solutions. *Id.* at 3:65-66. In other words, the design allows a system to forgo storing complete semantic routines in memory—as, for example, called for by the '520 patent. Instead, the system stores the target routines in memory and then quickly completes (*i.e.*, "patches") them each time an incompatible

instruction needs to be emulated, thereby saving storage space in memory. *Id.* at 367-4:4 ("By doing the instruction translation in a dynamic manner each time an incompatible instruction is encountered, it is unnecessary to save the translations for future use, saving the storage required to preserve these in target storage").

The concept of storing these half-completed target routines and completing them on the fly, however, was old hat at the time of the '261 patent. The advancement the '261 patent purported to add to the art was far smaller. As the patent examiner put it in the notice of allowance:

> The prior art of record does not teach associating one or more patching instructions with a target instruction in a target routine when the target instruction requires modification...***and not associating any patching instruction with any target not requiring modification.*** For these reasons claims 1 and 7 are deemed to be allowable over the prior art.

7/26/99 Notice of Allowance (Ex. 27) at 2.

Finally, before discussing the disputed terms, it is worth noting that the '261 and the '520 use slightly different terms to refer to the ***emulated*** system and the ***emulating*** system. In particular, the '261 uses the terms "incompatible" and "target" instead of the '520's "guest" and "host," respectively. This difference in terminology has no substantive import.

The parties dispute the construction of the terms "processor" (discussed above), "patching instruction," "incompatible instruction," "target instruction," and "target routine."

**1. "patching instruction" (claims 1, 2, 7, 8, and 10)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| An element of a target routine that is used to copy or modify some or all of a target instruction, to enable a guest instruction to be emulated. | An element of a target routine that is used to copy or modify some or all of a target instruction, to enable a guest instruction to be emulated in a dynamic manner each time a guest instruction is encountered. |

The only issue the Court needs to decide with respect to this claim term is whether the process of "patching" is done in a dynamic manner each time a guest instruction is encountered. The intrinsic evidence demonstrates that it is.

The express purpose of the '261 invention is to provide a system and method for creating semantic routines ***dynamically*** from incomplete "target routines." As the specification put it, "by doing the instruction translation ***in a dynamic manner*** each time an incompatible instruction is encountered, it is unnecessary to save the translations for future use." '261 at 3:67-4:4. The "SUMMARY OF THE INVENTION" section of the specification describes "patching" in "the subject invention" as follows:

> The subject invention described here uses an emulation preprocessing function, which decodes the incompatible instructions and, for each, determines the location of a target processor translation template routine that performs the function of the incompatible instruction....Also, the particular specifications of each incompatible instruction are extracted from the incompatible instruction, e.g. register numbers specified, ***dynamically***, each time an incompatible instruction is to be executed and used in target instructions of the template routine. ***It is not necessary to save previous translations to obtain good performance in the emulated execution***. The large amount of target storage required to save the existing translations for future execution and for the incompatible instruction directory is saved ***by the dynamic translation of incompatible instructions as they are encountered during emulated execution***.

*Id.* at 5:8-28 (emphasis added). Thus, in describing the "subject invention," the specification is unambiguous that the process of modifying the target routines to create semantic routines is done in a dynamic manner and each time an incompatible instruction is encountered.

And the specification is also unambiguous that it is this very process of modifying the target routines that the term "patching" was intended to encompass. As the specification put it, "[t]his process of modifying the target instructions in a target routine is herein termed 'patching.'" *Id.* at 8:50-57. This is straight-up lexicography. *See Sinorgchem Co. v. ITC*, 511 F.3d 1132, 1138 (Fed. Cir. 2007) ("We have frequently found that a definition set forth in the specification governs the meaning of the claims.").

In short, the "Summary of the Invention" section of the specification expressly defines "patching," repeatedly states that it is done "dynamically," and emphasizes that this dynamic manner is what distinguishes the "patching" disclosed in the patent from the prior art. "Patching instruction" should therefore be construed to include this "dynamic manner" limitation. *See Microsoft Corp. v. Multi-Tech Sys., Inc.*, 357 F.3d 1340, 1348 (Fed. Cir. 2004) (construing claims to be consistent with the specification's "Summary of the Invention"); *Alloc, Inc. v. ITC*, 342 F.3d 1361, 1368-69 (Fed. Cir. 2003) (relying on the patent specification's description of "the invention").

**2. "incompatible instruction " (claims 1, 7, 8, and 10) and "target instruction" (claims 1, 2, and 7-12)**

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| Incompatible Instruction - A ***language construct*** that specifies an operation and identifies its operands, if any, and ***pertains to the architecture*** being emulated, which is different than the architecture on which the emulator runs.<br><br>Target Instruction - A ***language construct*** that specifies an operation and identifies its operands, if any, and ***pertains to the architecture*** on which the emulator runs. | Incompatible Instruction - A ***string of digits*** that specifies an operation and identifies its operands, if any, and would be ***able to be directly executed by a processor*** of the emulated data processing system.<br><br>Target Instruction - A ***string of digits*** that specifies an operation and identifies its operands, if any, and ***can be directly executed by the processor*** to which it is directed. |

PSI agrees with IBM that the dispute over this term is directly related to the dispute over the meaning of the term "instruction." IBM's awkward "language construct" construction is contrary to the intrinsic and extrinsic evidence and should be rejected for the reasons articulated *supra* in Section III.B.

The two remaining issues the Court must decide are as follows: (1) whether an "incompatible instruction" can be directly executed by the processor of the emulated system; and (2) whether a "target instruction" can be directly executed by the processor of the target computer. In essence, both of these questions simply ask whether an instruction of a given ISA

(*e.g.*, an Intel x86 instruction) can be directly executed by a processor of that same ISA (*e.g.*, an Intel x86 processor). The intrinsic evidence, **which IBM completely ignores**, demonstrates that the answer is yes.

Independent claims 1 and 7 both establish that "incompatible instructions" and "target instructions" are directly "executable" by their respective "processors":

| Claim 1 | Claim 7 |
|---|---|
| 1. An emulation method for executing an incompatible computer program on a target processor, the incompatible program containing computer instructions natively ***executable on a different processor*** built to a computer architecture incompatible with target architecture used for building a target system containing the target processor, the target architecture defining target instructions ***executable on the target processor*** and the incompatible architecture defining execution results for the incompatible instructions, the emulation method comprising… | 7. An emulation method for executing an incompatible computer program on a target processor, the incompatible program containing computer instructions ***executable on a different processor*** built to a computer architecture incompatible with target architecture used for building a target system containing the target processor, the target architecture defining target instructions ***executable on the target processor*** and the incompatible architecture defining execution results for the incompatible instructions, the emulation method comprising… |

'261 at 19:23-29 & 20:63-21:2 (emphasis added). In other words, the claims explicitly say that "incompatible instructions" are things that can be directly executed by a processor of the emulated (*i.e.*, incompatible) system, while the target instructions are things that can be directly executed by the target processor.

IBM ignores this intrinsic evidence and, instead, makes two throw-away arguments. First, IBM argues that because (1) the parties agreed on definitions for "incompatible" and "target" and (2) IBM's construction simply combines those agreed definitions with IBM's construction for "instruction," that therefore (3) IBM's constructions for "incompatible instruction" and "target instruction" must be correct. But that argument, on its face, is a non sequitur: IBM's construction for "instruction" ***mistakenly excludes the fact that instructions can be directly executed by processors that share their ISA***. The fact that IBM added its tautological "pertains to…" language to its mistaken definition of "instruction"—*of course* an

incompatible instruction "pertains to" the incompatible architecture, and *of course* a target instruction "pertains to" the target architecture—does not change the fact that IBM's definition of "instruction" is wrong.

IBM's second argument is that PSI's construction relies on the definition of "machine instruction" and "**[n]othing in the intrinsic evidence references a machine instruction**." IBM Br. at 49 (emphasis added). This argument is simply ignorant of the intrinsic record:

> • In the Summary Of The Invention section the specification notes that the invention "allows a standard microprocessor to be used to perform ***target machine instructions*** that provide the same results as the incompatible instructions" '261 patent at 4:4-11 (emphasis added).
>
> • The specification notes that, for "complex ***incompatible machine instructions***...specifications in addition to the operation code may be used to determine what template routine is to be executed." *Id.* at 9:49-52 (emphasis added).
>
> • "Each target processor translation template routine is written with complete knowledge of the ***incompatible machine instruction*** whose function it will perform." *Id.* at 13:47-54 (emphasis added).
>
> • In its discussion of the prior art, the specification repeatedly refers to "machines instructions" when it describes emulation. *See id.* at 2:8-9 (discussing a related emulation technique for the "substitution of specific register values, and immediate displacement values, ***from source machine instruction into target machine instructions***"); 2:40-41 (describing emulation as a process in which "***target machine instructions*** [are] executed to provide the function of source ***machine instructions***.").

Thus, the specification repeatedly demonstrates that "target instructions" and "incompatible instructions" are both "machine instructions."

Furthermore, as noted above, the preambles for independent claims 1 and 7 both equate "incompatible instructions" with "***computer instructions*** natively executable on a different processor." *Id.* at 19:23-29 & 20:63-21:2 (emphasis added). In addition, claim 23 uses "the computer instruction" to refer back to the "target instruction" in claim 22. *Id.* at 24:19-46. These uses of "computer instruction" in the claims are significant because, according to IBM's own contemporaneous dictionary, "computer instruction" is ***synonymous*** with "machine instruction":

**computer instruction**: Synonym for machine instruction.

*IBM Dictionary of Computing* at 133. Thus, in light of the use of "computer instruction" in the claims and the repeated use of "machine instruction" in the specification, IBM's assertion that "[n]othing in the intrinsic evidence references a machine instruction" is simply wrong

The fact that the intrinsic evidence demonstrates that "target instructions" and "incompatible instructions" are both "machine instructions" supports PSI's construction: IBM's own dictionary states that a machine instruction "can be directly executed by a processor of a computer." *IBM Dictionary of Computing* at 408. And IBM itself has adopted the following construction for "machine instruction" in this lawsuit:

> A string of digits that specifies an operation and identifies its operands, if any, and can be directly executed by a processor of a computer.

IBM Br. at 54. Thus, the intrinsic record's repeated references to "machine instructions" establish that PSI's construction is correct.

Finally, it should be noted that IBM fails to provide any explanation for why it objects to PSI's "directly executed" construction. How can one possibly, in good faith, contest the fact that a target processor can directly execute target instructions, or that an "incompatible instruction" can be directly executed by a processor that shares its ISA? That is the same as saying that one objects to the statement "an Intel x86 processor can directly execute Intel x86 instructions." No reasonable person of ordinary skill in the art would object to that elementary assertion. IBM provides no explanation for its objection, and the only thing it is able to do with the intrinsic evidence is misrepresent it. PSI's construction should be adopted.

### 3. "target routine" (claims 1, 2, 7-9, 11, and 15)

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| The set or sequence of target instructions that enable an incompatible instruction to be run on target computer system. | A defined sequence of target instructions performing a function similar to, but not necessarily identically to, a corresponding incompatible |

| IBM's Proposed Construction | PSI's Proposed Construction |
|---|---|
| | instruction. |

There are two disputes between the parties relating to this claim term. The first has to do with what a target routine *is*—namely, whether or not it has an internal order. The second dispute concerns what a target routine *does*—namely, whether it "performs a function similar to, but not necessarily identically to, a corresponding incompatible instruction" ('261 patent at 19:35-37, *i.e.*, claim 1), as PSI contends, or whether it performs the broader function of "enabl[ing] an incompatible instruction to be run on a target computer system" (not in the intrinsic evidence), as IBM contends. PSI's construction is supported by the intrinsic and extrinsic evidence and should be adopted.

**a. Is a target routine a sequence or merely a set?**

Like a "semantic routine," a "target routine" is comprised of native instructions that are executed in order to emulate an incompatible instruction. '261 patent at claims 1 & 7. And a "target routine" must be a sequence for the same reasons that a semantic routine must be a sequence: the native instructions in any "target routine" have data and other dependencies that require that they be executed in order, because executing them out of order can give you different or nonsensical results (like, referring back to the morning routine example, putting your clothes on before you shower). All of the arguments PSI made with respect to why a "semantic routine" must be a sequence (*see supra* Section V.A.1) apply with equal force here, and PSI respectfully incorporates them by reference.

In addition, the fact that a "target routine" must be a sequence is embedded within the terms of the '261 claims. Claim 7, for example, refers to "executing the target instructions in a ***sequence*** indicated in the corresponding target routines." '261 at 21:30-32 (emphasis added). This claim language, which IBM ignores, establishes that a target routine cannot simply be an unordered set but is, instead, a "sequence."

Similarly, claim 18 calls for the step of "executing all target instructions in execution ***sequence*** in the target routines."[26] Again, the fact that there is an "execution sequence in the target routines" means that, as the term is used in the '261 patent, the target routine is ***not*** merely a set that can be executed in any order, but is instead a ***sequence*** that must be executed as such.

IBM's only argument to the contrary is to cite a single reference from the patent which states that "each incompatible instruction instance is translated to a set of equivalent target instruction in a corresponding target translation routine." IBM Br. at 50 (citing '261 at 9:27-30). But this proves nothing. First, as noted previously, a sequence is a special kind of set in the same way a square is a special kind of rectangle. Thus, nothing about the fact that the '261 talks about translating an incompatible instruction into a set of target instructions that are then put in a target routine means that the routine need not have any internal order. Second, as the quotes in the previous paragraphs make clear, the ***claims*** are explicit that the target routine ***does*** include a sequence. The single use of a more general word in the specification should not be read to imply the absence of the specific characteristic repeatedly called out by the claims.

Nor would it make sense to treat a target routine as a mere set of instructions to be performed in any order. The file history is explicit that the sole inventive portion of the patent is the notion that the target routines can sometimes be used *without* modification—namely, as ordinary semantic routines. *See* 7/26/1999 Notice of Allowance (Ex. 27) at 2. But, as discussed in the context of the '520 patent, semantic routines ***cannot*** simply be executed in any order because of the data and control dependencies involved. As the Court may recall from the earlier discussion, the inventor of the '520 patent expressly admitted this point in his deposition. *See* Mallick Depo. (Ex. 3) at 80:18-25; *see also* Patt Decl. ¶¶ 35-38.

[26] Although claim 18 has not been asserted, it should be treated as persuasive intrinsic evidence. *See Epcon Gas Sys. v. Bauer Compressors, Inc.*, 279 F.3d 1022, 1030 (Fed. Cir. 2002) ("the same term or phrase should be interpreted consistently where it appears in claims of common ancestry.").